Posgrado en Ciencias y Tecnologías de la Información

UAM Iztapalapa

Inteligencia Artificial

Seguimiento por filtro de partículas

Proyecto Final

Muñoz Morales José Alberto

Andrés Eduardo Flores Roque

Francisco Javier Cruz González

Introducción

En este trabajo final, correspondiente a la UEA de Inteligencia Artificial de la Maestría en Ciencias y Tecnologías de la Información, se desarrolla un algoritmo en lenguaje Python basado en el filtro de partículas. Para cumplir con este objetivo, el Dr. Rene Mac Kinney Romero proporcionó un video en formato .mp4, en el que se observa a una mujer caminando con ropa azul.

Pasos del algoritmo de filtro de partículas:

Inicialización: Comenzamos generando un conjunto de partículas distribuidas aleatoriamente en la imagen, cada uno representando una posible hipótesis de la posición del objeto.

Predicción (Propagación): A medida que el sistema evoluciona, las partículas se mueven de acuerdo con las dinámicas del sistema, y se predice el estado futuro de cada partícula (en este caso, puede ser un pequeño desplazamiento aleatorio para reflejar la incertidumbre del movimiento del objeto).

Medición: Comprobamos cuán "probable" es que cada partícula corresponda al objeto en el fotograma actual, comparando el color de la región alrededor de cada partícula con el color esperado del objeto.

Actualización de pesos: Calculamos un peso para cada partícula basado en la probabilidad de que esa partícula esté cerca del objeto (calculando la probabilidad de pertenencia de los píxeles en la vecindad de cada partícula), es decir las partículas se actualizan utilizando las mediciones u observaciones más recientes, evaluando cuan probable es cada partícula con respecto a los datos observados.

Re-mestreo: Las partículas con pesos bajos (menos probables) se eliminan, y se generan nuevas partículas basadas en las que tienen pesos más altos (más probables). Es decir, se hace una selección, esto ayuda a evitar que partículas irrelevantes dominen el proceso. [4]

Estimación del estado: Después de una serie de iteraciones, el estado estimado del sistema puede obtenerse como una media ponderada de las partículas, donde los pesos reflejan la probabilidad de cada estado. Es decir, la posición estimada del objeto es calculada como la media ponderada de las posiciones de todas las partículas.

Proceso de construcción del algoritmo

El objetivo del sistema es seguir un objeto en movimiento en video según el color "azul". Para lograrlo, es esencial entender cómo se representa la información del video.

Un video consiste en una secuencia de imágenes (fotogramas), mostradas a unos 30 fotogramas por segundo. Cada fotograma está formado por píxeles, y cada píxel tiene un color representado en el modelo RGB (Rojo, Verde, Azul). Cada componente de color (R, G, B) se codifica generalmente en un byte (8 bits), con valores de 0 a 255. Así, un color se describe por tres valores: "R", "G" y "B".

Debido a que los valores de cada fotograma varían de forma impredecible, podemos tratarlos como variables aleatorias. Estas variables nos ayudarán a definir el espacio muestral, calcular las probabilidades asociadas y su función de distribución acumulada, lo que permitirá asignar pesos a las partículas para el seguimiento del objeto.

Trabajamos en formato HSV en lugar del formato RGB:

El formato HSV (Matiz, Saturación, Valor) es útil para detectar y seguir objetos por su color, ya que el matiz (H) identifica los colores más fácilmente que el RGB. [5]

- Matiz (H - [0°, 180°]): Representa el color. Para tonos azules, el rango está entre 100° y 140°. Se recomienda permitir un rango de ±15° a ±30° para cubrir variaciones.
- Saturación (S - [0, 255]): Muestra la intensidad del color. Un rango de 100 a 255 es útil para adaptarse a variaciones por iluminación.
- Valor (V - [0, 255]): Indica la luminosidad del color. Un rango de 50 a 255 cubre las sombras o baja iluminación.

Definición de un rango de tolerancia para cada componente en HSV

Para un tono azul.

- Matiz: 111° ± 20° (es decir, entre 91° y 131°).
- Saturación: 214 ± 50 (por ejemplo, entre 164 y 255).
- Valor: 118 ± 50 (por ejemplo, entre 68 y 168).

Los rangos propuestos parecen razonables para detectar un objeto “azul”. Sin embargo, es importante verificar experimentalmente que estos rangos se mantengan consistentes en diferentes condiciones de iluminación y variabilidad del color azul en la escena.

Fórmula de comparación

Una vez definido estos rangos, comparamos un color con el objetivo verificando si sus componentes HSV están dentro de los rangos definidos para cada componente (H, S, V), es un enfoque para convertir los valores de los píxeles en eventos binarios (pertenencia o no pertenencia). Ahora podemos usar esta función para **evaluar la probabilidad** de que un píxel o una partícula pertenezca a un color "azul".

Definamos la función de pertenencia para H, S y V, con parámetro del rango que devolverá 0 o 1.

- Para H: Si H está dentro del rango regresamos 1 en otro caso 0.
- Para S: Si S está dentro del rango regresamos 1 en otro caso 0.
- Para V: Si V está dentro del rango regresamos 1 en otro caso 0.

Entonces, tenemos un espacio muestral de 8 (2^3) elementos. La variable aleatoria, que es una función, la construimos con estas combinaciones, en este caso corresponde a cada combinación posible. Si este es su domino entonces deberá tener una imagen, y la relación es el numero de unos que tiene la combinación.

Entonces va(0,0,0) es 0, va(0,0,1), va(0,1,0) y va(1,0,0) valen 1, va(0,1,1)=2, va(1,0,1), va(1,0,1) valen 2 y va(1,1,1)=3.

Con esto hemos definido nuestra variable aleatoria x que representa cuántos componentes (de H, S y V) están dentro de su rango, porque los pixeles o partículas son impredecibles e independientes.

Es decir:

- x=0 si ninguno de los componentes está dentro de su rango.
- x=1 si uno de los componentes está dentro de su rango.
- x=2 si dos de los componentes están dentro de su rango.
- x=3 si los tres componentes están dentro de su rango.

Definamos nuestra función de probabilidad de variable aleatoria x, su dominio es 0, 1, 2 y 3, que corresponden a la imagen de nuestra función aleatoria que definimos antes. Como es una función de probabilidad f(x), debemos de determinar cuál es la probabilidad (casos favorables / casos posibles).

- La probabilidad de 0 unos es 1/8 (1 combinación / 8 combinaciones).
- La probabilidad de 1 uno es 3/8 (3 combinaciones / 8 combinaciones).
- La probabilidad de 2 unos es 3/8 (3 combinaciones / 8 combinaciones).
- Y finalmente 3 unos es 1/8 (1 combinación / 8 combinaciones).
- La suma de las probabilidades es 1.

La función de distribución acumulada CDF es la suma acumulada de las probabilidades $F(x) = P(X \leq x)$. Dado que tenemos 4 posibles valores para X (0, 1, 2, 3), la CDF será es:

- F (0) = P(X=0) = 1/8, probabilidad de que una partícula no coincida con los rangos de color.
- F (1) = P(X=0) + P(X=1) = 1/8 + 3/8 = 4/8=1/2, acumula la probabilidad de que la partícula tenga uno de los tres componentes dentro de los rangos.
- F (2) = P(X=0) + P(X=1) + P(X=2) = 1/8 + 3/8 + 3/8 = 7/8, acumula la probabilidad de que la partícula tenga dos componentes dentro de los rangos.
- F (3) = P(X=0) + P(X=1) + P(X=2) + P(X=3) = 1/8 + 3/8 + 3/8 + 1/8 = 1, es la probabilidad de que todos los componentes coincidan con los rangos definidos.

El re-muestreo es una parte clave del filtro de partículas, En cada estado, las partículas con mayor probabilidad tienen más probabilidades de ser seleccionadas para representar el estado del sistema en el siguiente paso. Esto lo haremos utilizando el peso de cada partícula, y utilizaremos la CDF para asignar este peso a cada articula, ya que cuanto más se acerque al color azul, mayor será el peso de la partícula (F(3) = 1) porque representaría la probabilidad acumulada de que una partícula pertenezca a un color dentro de los rangos especificados en este caso el azul.

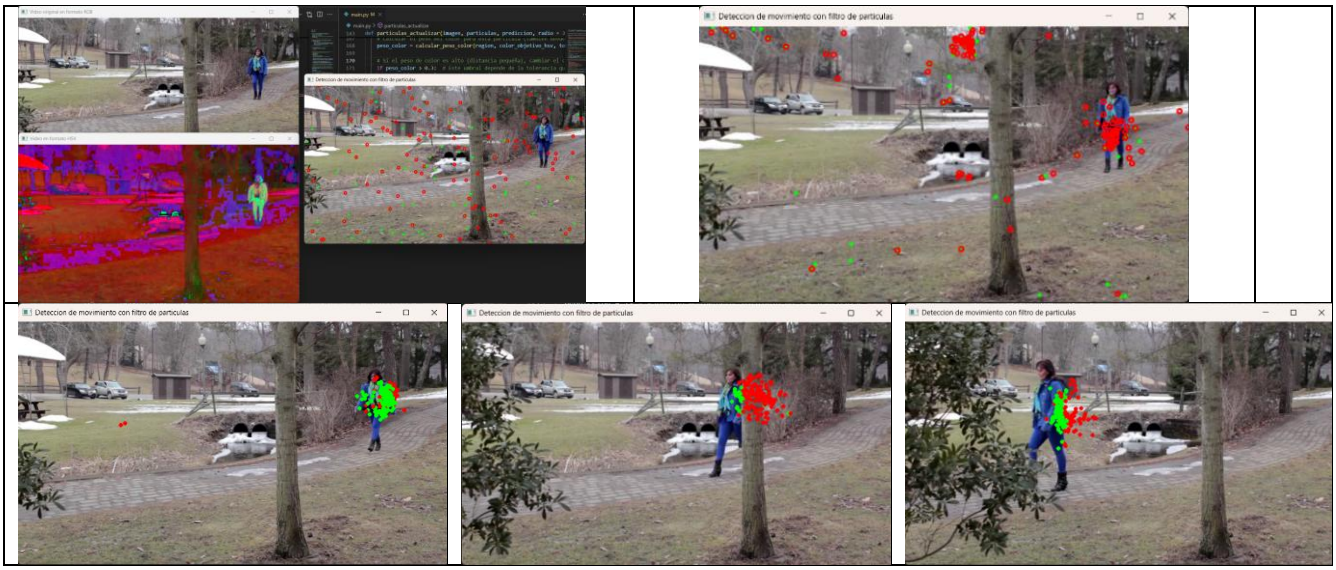
Pseudocódigo del algoritmo general:

```
Funcion posicion_particulas <- inicializar_particulas ( num_particulas, imagen_hsv_shape )
        // generar las posiciones de las particulas
        // de manera uniforme en toda la imagen
Fin Funcion

Funcion re_particulas <- remuestrear_particulas ( particulas, pesos )
```

```
        // seleccion de las particulas mas probables
Fin Funcion

Funcion nuevos_pesos <- actualizar_pesos ( partiulas, imagen_hsv )
        // asignacion de pesos de acuerdo con lo observado
Fin Funcion

Funcion propagar_particulas <- propagar_particulas ( particulas,  imagen_hsv_shape,  prediccion )
        // mover las particulas
Fin Funcion

Algoritmo filtro_particulas
  captura = captura_video
  particulas<-vacio
  num_particulas<-0
        Mientras captura_esta_abierta Hacer
                frame<-captura_leer_frame
                imagen_hsv<-convertir_a_formato_HSV
                Si len_particulas=0 Entonces
                        particulas<-inicializar_particulas(num_particulas, imagen_hsv_shape)
                Fin Si
                particulas<-propagar_particulas(particulas, imagen_hsv_shape, prediccion=5)
                pesos<-actualizar_pesos(particulas, imagen_hsv)
                particulas<-remuestrear_particulas(particulas, pesos)
                Escribir particulas_imagen
        Fin Mientras
FinAlgoritmo
```

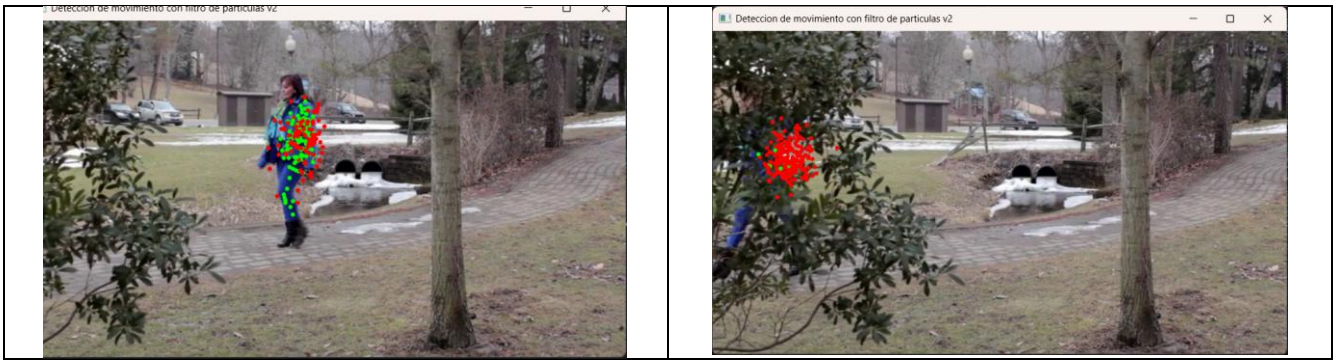
Resultado de la primera versión:

En las siguientes imágenes muestran que tiene buena respuesta el sistema:



Del análisis anterior no damos cuenta que se puede mejorar, ya que la clave está en seleccionar una función que me asigne pesos, en la siguiente versión, ahora se tomara en cuenta la distancia euclidiana, y esta distancia pasara por un función exponencial decreciente, ya que esto me garantiza la un valor alto a distancias cercanas a cero y suavizando la caída distancias más grandes, en formato HSV y la distancia máxima es de aproximadamente 402, así que para la versión 2 del algoritmo se tiene una tolerancia de 70, que corresponderá a un peso de 0.5 y a partir de ahí el peso es mucho menor.

Con esta nueva versión se tiene un seguimiento mucho más rápido y preciso.



Conclusiones

En este trabajo se desarrolló e implementó un algoritmo de filtro de partículas para realizar el seguimiento dinámico de un color objetivo en un video de entrada, demostrando su efectividad en entornos dinámicos.

Se desarrollaron dos versiones del algoritmo. En la primera, los pesos de las partículas se calcularon exclusivamente con base en una distribución de valores específicos; mientras que, en la segunda versión, se introdujo un enfoque adaptativo mediante una función exponencial decreciente, que dependía de la distancia euclidiana entre el color promedio de cada región asignada a las partículas y el color objetivo. Esta mejora permitió que las partículas se agruparan de manera más precisa en torno a la región del color especificado, definido como una tonalidad azul en el espacio HSV.

El sistema mostró una alta capacidad para mantener la cohesión de las partículas alrededor de la región objetivo. Los resultados obtenidos destacan que la combinación de técnicas probabilísticas para la inicialización y esquemas adaptativos para la ponderación es una estrategia eficiente para el seguimiento visual de características específicas en videos en movimiento.

Referencias

[1] J. Elfring, E. Torta and R. Van de Molengraft. “Particle Filters: A Hands-On Tutorial”. *Sensors Basel*. June 9, 2021.

[2] P. Abdeel. “Particle Filters”. UC Berkeley EECS. (https://homes.cs.washington.edu/~todorov/courses/cseP590/16_ParticleFilter.pdf)

[3] 3. Blanco-Claraco J.L., Mañas-Alvarez F., Torres-Moreno J.L., Rodriguez F., Gimenez-Fernandez A. Benchmarking Particle Filter Algorithms for Efficient Velodyne-Based Vehicle Localization. Sensors. 2019;19:3155. doi: 10.3390/s19143155.

[4] Russell, S. J., & Norvig, P. (2020). *Inteligencia artificial: Un enfoque moderno* (2ª ed.). Pearson.

[5] EcuRed. (2025). *Modelo HSV*. EcuRed. https://www.ecured.cu/Modelo_HSV

Anexo 1

Código del filtro de partículas, (dos versiones por separado):

GitHub: https://github.com/fjcg-mx/filtro-particulas